Projet: Application de Gestion de Restaurant

Le restaurant

Archi'Burger : un restaurant spécialisé dans les burgers aux noms de styles architecturaux. Ouverture du mardi au dimanche de 11 H 30 à 22 H 30.

Les Menus :

Le Corbusier :

Burger au choix : L'Art Déco, Le contemporain
Frites
Boisson au choix : eau, coca, fanta, ice tea
Prix : 18,5 €

Gustave Eiffel :

Burger au choix : L'industriel, Le brutaliste
Frites
Boisson au choix : eau, coca, fanta, ice tea
Prix : 16.5 €

Eugène Haussmann :

Burger au choix : Le classique, Le victorien
Frites
Boisson au choix : eau, coca, fanta, ice tea
Prix : 14.5 €

Les burgers :

- **Le classique**

  Description : Un burger intemporel avec un steak juteux, du cheddar fondant, de la laitue croquante, des tomates fraîches et une sauce maison. Simple, efficace, indémodable… comme les fondations de toute bonne architecture.
  Prix : 11 €

- **Le victorien :**

  Description : Un burger raffiné aux saveurs riches : steak grillé, bacon croustillant, oignons caramélisés, cheddar vieilli, roquette, et une sauce légèrement sucrée au miel et à la moutarde. Un clin d'œil au charme ancien et à la sophistication des maisons victoriennes.
  Prix : 11 €

- **L'industriel**

  Description : Un burger brut et costaud : double steak, cheddar fumé, pickles, oignons rouges, sauce barbecue épicée, et pain aux céréales. Un design robuste et sans fioritures, comme une usine transformée en loft urbain
  Prix : 13 €

- **Le brutaliste**

  Description : Un burger massif et minimaliste : steak XL, fromage fondu en abondance, champignons poêlés, pain noir, et une sauce poivre intense. Des saveurs franches et puissantes, à l'image du béton brut qui inspire ce style.

  Prix : 13 €

- **L'Art Déco**

  Description : Un burger audacieux et élégant : steak, fromage bleu, poire grillée, noix concassées, et une touche de roquette. Une composition géométrique et surprenante qui rappelle le luxe et l'excentricité des années 20.

  Prix : 15 €

- **Le contemporain**

  Description : Un burger moderne et équilibré : steak végétal ou bœuf au choix, avocat, tomates séchées, feta, pousses d'épinards et sauce yaourt-citron. Léger, design, et dans l'air du temps — pour les palais tournés vers l'avenir.

  Prix : 15 €

- **Vous allez définir les différentes étapes pour mener à bien ce projet.**

- **- Par quoi allez-vous commencer ?**

  → Trello pour s'organiser
  → Analyse du besoin client, recueil d'informations
  → Cahier des charges
  → Choix de l'architecture
  → Etude Merise pour concevoir la BDD
  → Diagramme de classes UML
  → Développement du projet

- - Quelle architecture allez-vous adopter pour ce projet et pourquoi ?

Architecture multicouches:

-Couche Interface(FrontEnd)

-Couche Domaine (core)(BackEnd)

-Couche Infrastructure(BDD)

Justification :

-Le découpage en couches permet une séparation claire des responsabilités

-Les règles métier sont isolées et indépendantes des détails d'implémentation

-Facilite la testabilité de chaque couche séparément

-L'application peut évoluer plus facilement (changement de base de données, d'interface utilisateur)

-Permet l'application des principes SOLID

- **- Quel paradigme : Structurel (procédural), POO ou Fonctionnelle ? Pourquoi ?**

  Programmation Orientée Objet (POO):

  -Le domaine se modélise naturellement en objets (Réservation, Plat, Commande)

  -Permet l'encapsulation des données et des comportements

  -Facilite l'application des design patterns

  -Supporte bien l'évolution du système par extension plutôt que modification (principe Open/Closed)

  -Plus adapté que le paradigme procédural pour gérer la complexité des interactions entre entités

  -Plus maintenable que le paradigme fonctionnel pour ce type d'application avec état

- - Allez-vous utiliser un design pattern ? Si oui, lequel ?

  → **DAO Pattern**

  Objectif:

Le pattern DAO (Data Access Object) vise à séparer la logique d'accès aux données de la logique métier. Il fournit une interface abstraite pour accéder à une source de données (base de données, service web, fichier, etc.) indépendamment de son implémentation technique.

Application dans le projet :

Séparer l'accès aux données des entités métier

Isoler le code SQL ou requêtes de base de données du reste de l'application

Faciliter le changement de technologie de persistance sans impacter le reste du code

Avantages pour le projet :

Séparation des responsabilités

Abstraction de la source de données

Centralisation de la logique d'accès aux données

→ **Repository Pattern**

Objectif : Abstraire la logique d'accès aux données et fournir des méthodes pour obtenir des objets de domaine.

Application dans le projet :

Sépare la logique métier de la logique d'accès aux données

Permet de changer facilement le mécanisme de stockage

Avantages pour le projet :

Facilite les tests unitaires en permettant de mocker les repositories

Centralise la logique d'accès aux données

Améliorer la maintenabilité en isolant les changements de la couche de persistance